UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/708,722	11/09/2000	Stephan J. Jourdan	2207/9800	2194

25693 7590 04/18/2007

KENYON & KENYON LLP
RIVERPARK TOWERS, SUITE 600
333 W. SAN CARLOS ST.
SAN JOSE, CA 95110

EXAMINER	
LI, AIMEE J	
ART UNIT	**PAPER NUMBER**
2183	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	04/18/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	**Application No.** 09/708,722	**Applicant(s)** JOURDAN ET AL.
	Examiner Aimee J. Li	**Art Unit** 2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *18 December 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-19* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-19 have been considered.

2. In view of the Appeal Brief filed on 18 December 2006, PROSECUTION IS HEREBY REOPENED. The new rejection is set forth below.

3. To avoid abandonment of the application, appellant must exercise one of the following two options:

(1) file a reply under 37 CFR 1.111 (if this Office action is non-final) or a reply under 37 CFR 1.113 (if this Office action is final); or,

(2) initiate a new appeal by filing a notice of appeal under 37 CFR 41.31 followed by an appeal brief under 37 CFR 41.37. The previously paid notice of appeal fee and appeal brief fee can be applied to the new appeal. If, however, the appeal fees set forth in 37 CFR 41.20 have been increased since they were previously paid, then appellant must pay the difference between the increased fees and the amount previously paid.

4. A Supervisory Patent Examiner (SPE) has approved of reopening prosecution by signing below.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 1-5 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims 1-5 contain non-functional descriptive material, since all that is claimed is the compilation and order of data, neither of which are functional descriptive material. Functional descriptive material must impart functionality when employed as a

computer components. These claims contain nothing of this matter. They merely claim a compilation of data arranged in reverse program order. The "cache line" and "cache entries" recitations for storing the data is merely an intended use recitation and does nothing to further limit the claim to or show any type of functional descriptive material.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claims 6-7 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 6 recites "the segment cache of claim 5 included therein…". It is unclear whether this claim is a dependent claim on claim 5 or intended to be an independent claim. If the claim is intended to be independent please replace "the segment cache of claim 5 included there" with the claim language in claim 5 to clarify. If the claim is meant to be dependent, please preface the claim with language such as, "The segment cache of claim 5…" or the likes.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

10. Claims 1-5 and 8-14 are rejected under 35 U.S.C. 102(e) as being taught by Kyker et al., U.S. Patent Number 6,578,138 (herein referred to as Kyker).

11. Referring to claim 1, Kyker has taught a cache comprising:

a. A cache line to store an instruction segment further comprising a plurality of instructions stored in sequential positions of cache line in reverse program order (Kyker Abstract "...a cache unit, which includes a data array that stores traces...In one exemplary method of unrolling loops, the processor or trace cache unrolls loops..."; column 2, line 59 to column 3, line 33 "...when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop until the trace is a minimal length...In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long..."; Figure 1; and Figure 2 – In regards to Kyker, as shown in Figure 2 and explained the in the cited lines, the loop has three main instructions and, when the backwards branch is encountered, the program jumps backwards, i.e. reverses directions, to the beginning of the loop instead of moving forward in the program. As shown in Figure 2, when L_H is encountered, previously stored instructions L_2 and L_3 are stored again in consecutive lines in the cache, e.g. the trace cache reverses the program direction and stores previously stored instructions again in the trace cache when a loop is encountered.).

12. Referring to claim 2, Kyker has taught the cache of claim 1, wherein the instruction segment is an extended block (Kyker column 1, lines 30-45 "...the target address, the backward

taken branch, and any micro-ops between the two form a loop…"; column 2, line 59 to column 3, line 33 "…Exemplary trace T1 includes a total of four micro-ops…The second exemplary trace T2 includes the same loop, L_H, L_1, L_2, but does not include any micro-op preceding the loop itself…"; Figure 1; and Figure 2 – In regards to Kyker, the loop contains a plurality of instructions, i.e. block of instructions, with four instructions and includes the extra micro-op preceding the loop itself.).

13. Referring to claim 3, Kyker has taught the cache of claim 1, wherein the instruction segment is a trace (Kyker Abstract "…a cache unit, which includes a data array that stores traces…In one exemplary method of unrolling loops, the processor or trace cache unrolls loops…"; column 2, line 59 to column 3, line 33 "…when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop until the trace is a minimal length…In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long…"; Figure 1; and Figure 2).

14. Referring to claim 4, Kyker has taught the cache of claim 1, wherein the instruction segment is a basic block (Kyker column 1, lines 30-45 "…the target address, the backward taken branch, and any micro-ops between the two form a loop…"; column 2, line 59 to column 3, line 33 "…Exemplary trace T1 includes a total of four micro-ops…The second exemplary trace T2 includes the same loop, L_H, L_1, L_2, but does not include any micro-op preceding the loop itself…"; Figure 1; and Figure 2 – In regards to Kyker, the loop contains a plurality of instructions, i.e. block of instructions, with three instructions and does not include the extra micro-op preceding the loop itself.).

15. Referring to claim 5, Kyker has taught a segment cache for a front-end system in a processor (Kyker column 5, line 45 to column 6, line 2 "…The trace cache also includes, for example, a control bloc **109** and a instruction decoder **111**..."; Figure 10; and Figure 11 – In regards to Kyker, the trace cache stores portions, e.g. segments, of the program and, as shown in Figures 10 and 11, is part of the system associated with decoding instructions, which is prior to execution and retirement of instructions, i.e. the back-end system.), comprising a plurality of cache entries to store instructions of instruction segments in reverse program order (Kyker Abstract "…a cache unit, which includes a data array that stores traces…In one exemplary method of unrolling loops, the processor or trace cache unrolls loops…"; column 2, line 59 to column 3, line 33 "…when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop until the trace is a minimal length…In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long…"; Figure 1; and Figure 2 – In regards to Kyker, as shown in Figure 2 and explained the in the cited lines, the loop has three main instructions and, when the backwards branch is encountered, the program jumps backwards, i.e. reverses directions, to the beginning of the loop instead of moving forward in the program. As shown in Figure 2, when L_H is encountered, previously stored instructions L_2 and L_3 are stored again in consecutive lines in the cache, e.g. the trace cache reverses the program direction and stores previously stored instructions again in the trace cache when a loop is encountered.).

16. Referring to claim 8, Kyker has taught a method comprising:

a. Building an instruction segment based on program flow (Kyker Abstract "…a cache unit, which includes a data array that stores traces…In one exemplary

method of unrolling loops, the processor or trace cache unrolls loops…"; column 2, line 59 to column 3, line 33 "…when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop until the trace is a minimal length…In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long…"; Figure 1; and Figure 2), and

b. Storing instructions of the instruction segment in a cache entry in reverse program order (Kyker Abstract "…a cache unit, which includes a data array that stores traces…In one exemplary method of unrolling loops, the processor or trace cache unrolls loops…"; column 2, line 59 to column 3, line 33 "…when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop until the trace is a minimal length…In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long…"; Figure 1; and Figure 2 – In regards to Kyker, as shown in Figure 2 and explained the in the cited lines, the loop has three main instructions and, when the backwards branch is encountered, the program jumps backwards, i.e. reverses directions, to the beginning of the loop instead of moving forward in the program. As shown in Figure 2, when L_H is encountered, previously stored instructions L_2 and L_3 are stored again in consecutive lines in the cache, e.g. the trace cache reverses the program direction and stores previously stored instructions again in the trace cache when a loop is encountered.).

17. Referring to claim 9, Kyker has taught the method of claim 8, further comprising:

a. Building a second instruction segment based on program flow, and if the first and second instruction segments, overlap, extending the first instruction segment to include non-overlapping instructions from the second instruction segment (Kyker Abstract "...a cache unit, which includes a data array that stores traces...In one exemplary method of unrolling loops, the processor or trace cache unrolls loops..."; column 2, line 59 to column 3, line 33 "...when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop until the trace is a minimal length...In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long..."; Figure 1; and Figure 2 – In regards to Kyker, as shown in Figure 1, the trace is built as normal, e.g. a first instruction segment is built normally, when there is no backward-taken branch. When a backward branch is taken to form another, second instruction segment, another iteration of the loop is added to the trace, thereby extending the first segment to include the first segment.).

18. Referring to claim 10, Kyker has taught the method of claim 9, wherein the extending comprises storing the non-overlapping instructions in the cache in reverse program order in successive cache positions adjacent to the instructions from the first instruction segment (Kyker Abstract "...a cache unit, which includes a data array that stores traces...In one exemplary method of unrolling loops, the processor or trace cache unrolls loops..."; column 2, line 59 to column 3, line 33 "...when the trace cache determines that a loop is present, the trace cache

continues to build the trace by building additional iterations of the loop until the trace is a minimal length…In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long…"; Figure 1; and Figure 2 – In regards to Kyker, as shown in Figure 2 and explained the in the cited lines, the loop has three main instructions and, when the backwards branch is encountered, the program jumps backwards, i.e. reverses directions, to the beginning of the loop instead of moving forward in the program. As shown in Figure 2, when L_H is encountered, previously stored instructions L_2 and L_3 are stored again in consecutive lines in the cache, e.g. the trace cache reverses the program direction and stores previously stored instructions again in the trace cache when a loop is encountered.).

19. Referring to claim 11, Kyker has taught the method of claim 8, wherein the instruction segment is an extended block (Kyker column 1, lines 30-45 "…the target address, the backward taken branch, and any micro-ops between the two form a loop…"; column 2, line 59 to column 3, line 33 "…Exemplary trace T1 includes a total of four micro-ops…The second exemplary trace T2 includes the same loop, L_H, L_1, L_2, but does not include any micro-op preceding the loop itself…"; Figure 1; and Figure 2 – In regards to Kyker, the loop contains a plurality of instructions, i.e. block of instructions, with four instructions and includes the extra micro-op preceding the loop itself.).

20. Referring to claim 12, Kyker has taught the method of claim 8, wherein the instruction segment is a trace (Kyker Abstract "…a cache unit, which includes a data array that stores traces…In one exemplary method of unrolling loops, the processor or trace cache unrolls loops…"; column 2, line 59 to column 3, line 33 "…when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop

until the trace is a minimal length…In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long…"; Figure 1; and Figure 2).

21. Referring to claim 13, Kyker has taught the method of claim 8, wherein the instruction segment is a basic block (Kyker column 1, lines 30-45 "…the target address, the backward taken branch, and any micro-ops between the two form a loop…"; column 2, line 59 to column 3, line 33 "…Exemplary trace T1 includes a total of four micro-ops…The second exemplary trace T2 includes the same loop, L_H, L_I, L_2, but does not include any micro-op preceding the loop itself…"; Figure 1; and Figure 2 – In regards to Kyker, the loop contains a plurality of instructions, i.e. block of instructions, with three instructions and does not include the extra micro-op preceding the loop itself.).

22. Referring to claim 14, Kyker has taught a processing engine, comprising:

a. A front end stage to build and store instruction segments (Kyker column 5, line 45 to column 6, line 2 "…The trace cache also includes, for example, a control bloc **109** and a instruction decoder **111**..."; Figure 10; and Figure 11 – In regards to Kyker, the trace cache stores portions, e.g. segments, of the program and, as shown in Figures 10 and 11, is part of the system associated with decoding instructions, which is prior to execution and retirement of instructions, i.e. the back-end system.), instructions provided therein in reverse program order (Kyker Abstract "…a cache unit, which includes a data array that stores traces…In one exemplary method of unrolling loops, the processor or trace cache unrolls loops…"; column 2, line 59 to column 3, line 33 "…when the trace cache determines that a loop is present, the trace cache continues to build the trace by

building additional iterations of the loop until the trace is a minimal length...In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long..."; Figure 1; and Figure 2 – In regards to Kyker, as shown in Figure 2 and explained the in the cited lines, the loop has three main instructions and, when the backwards branch is encountered, the program jumps backwards, i.e. reverses directions, to the beginning of the loop instead of moving forward in the program. As shown in Figure 2, when L_H is encountered, previously stored instructions L_2 and L_3 are stored again in consecutive lines in the cache, e.g. the trace cache reverses the program direction and stores previously stored instructions again in the trace cache when a loop is encountered.), and

b. An execution unit in communication with the front end stage (Kyker column 5, line 61 to column 6, line 2 "...the computer system includes a processor **121**, a trace cache **101**, and a computer memory **123**..." and Figure 11).

Claim Rejections - 35 USC § 103

23. Claims 6-7 and 15-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kyker et al., U.S. Patent Number 6,578,138 (herein referred to as Kyker) as applied to claims 5 and 14 above, and further in view of Rotenberg et al.'s "A Trace Cache Microarchitecture and Evaluation" IEEE ©1999 (herein referred to as Rotenberg).

24. Referring to claim 6, Kyker has taught apparatus comprising;

a. An instruction cache system (Kyker column 5, lines 45-60 "...an exemplary embodiment of a trace cache..." and Figure 10),

b. An instruction segment system (Kyker column 5, line 45 to column 6, line 2 "…The trace cache also includes, for example, a control block **109** and a instruction decoder **111**..."; Figure 10; and Figure 11 – In regards to Kyker, the trace cache stores portions, e.g. segments, of the program.), comprising:

i. A fill unit provided in communication with the instruction cache system (Kyker column 5, lines 45-60 "…the control block **109** facilitates the build process by providing write enable commands to the data array…" and Figure 10),

ii. The segment cache of claim 5 included therein (See the above rejection of claim 5), and

25. Kyker has not taught a selector coupled to an output of the instruction cache system and to an output of the segment cache. Rotenberg has taught a selector coupled to an output of the instruction cache system and to an output of the segment cache (Rotenberg Section 2.1 Trace-Level Sequencing "…The output of the trace cache is one or more traces…"; Section 2.2 Instruction-Level Sequencing "The *outstanding trace buffers* in Fig. 2 are used to 1) construct new traces that are not in the trace cache and 2) track branch outcomes…"; and Figure 2 – In regards to Rotenberg, Figure 2 shows the output of the trace cache and the outstanding trace buffers are connected to a line that has two inputs and one output, which is a selector. The selector chooses between and existing trace in the trace cache, i.e. a trace cache hit, or a newly formed trace, i.e. a trace cache miss or misprediction.) A person of ordinary skill in the art at the time the invention was made, and as taught by Rotenberg, would have recognized that the trace cache system provides fast trace-level sequencing while providing a method to create nonexistent

traces or repair mispredicted traces (Rotenberg Section 2.1 Trace-Level Sequencing "...The trace predictor and trace cache together provide fast trace-level sequencing...Instruction-level sequencing, discussed in the next section, is required to construct nonexistent traces or repair trace mispredictions."). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the trace caches, buffers, and selector of Rotenberg in the device of Kyker to increase the speed of trace-level sequencing while using instruction-level sequencing to create new or correct mispredicted traces.

26. Referring to claim 7, Kyker has not taught an apparatus of claim 6, wherein the instruction segment system further comprises a segment predictor provided in communication with the segment cache. Rotenberg has taught an apparatus of claim 6, wherein the instruction segment system further comprises a segment predictor provided in communication with the segment cache (Rotenberg Section 2.1 Trace-Level Sequencing "...A *next trace predictor*[14] treats traces as basic units and explicitly predicts sequences of traces..." and Figures 2). A person of ordinary skill in the art at the time the invention was made, and as taught by Rotenberg, would have recognized that the trace predictor achieves high branch prediction throughput with a single prediction per cycle (Rotenberg Section 2.1 Trace-Level Sequencing "...high branch prediction throughput is implicitly achieved with only a single trace prediction per cycle..."). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the trace predictor of Rotenberg in the device of Kyker to achieve high branch prediction throughput in a single prediction cycle.

27. Referring to claim 15, Kyker has taught the processing engine of claim 14, wherein the front-end stage comprises:

a. An instruction cache system (Kyker column 5, lines 45-60 "...an exemplary embodiment of a trace cache..." and Figure 10),

b. An instruction segment system (Kyker column 5, line 45 to column 6, line 2 "...The trace cache also includes, for example, a control bloc **109** and a instruction decoder **111**..."; Figure 10; and Figure 11 – In regards to Kyker, the trace cache stores portions, e.g. segments, of the program.), comprising:

i. A fill unit provided in communication with the instruction cache system (Kyker column 5, lines 45-60 "...the control block **109** facilitates the build process by providing write enable commands to the data array..." and Figure 10),

ii. A segment cache (Kyker column 5, line 45 to column 6, line 2 "...The trace cache also includes, for example, a control bloc **109** and a instruction decoder **111**..."; Figure 10; and Figure 11 – In regards to Kyker, the trace cache stores portions, e.g. segments, of the program.), and

28. Kyker has not taught a. selector coupled to an output of the instruction cache system and to an output of the segment cache. Rotenberg has taught a selector coupled to an output of the instruction cache system and to an output of the segment cache (Rotenberg Section 2.1 Trace-Level Sequencing "...The output of the trace cache is one or more traces..."; Section 2.2 Instruction-Level Sequencing "The *outstanding trace buffers* in Fig. 2 are used to 1) construct new traces that are not in the trace cache and 2) track branch outcomes..."; and Figure 2 – In regards to Rotenberg, Figure 2 shows the output of the trace cache and the outstanding trace buffers are connected to a line that has two inputs and one output, which is a selector. The

selector chooses between and existing trace in the trace cache, i.e. a trace cache hit, or a newly formed trace, i.e. a trace cache miss or misprediction.) A person of ordinary skill in the art at the time the invention was made, and as taught by Rotenberg, would have recognized that the trace cache system provides fast trace-level sequencing while providing a method to create nonexistent traces or repair mispredicted traces (Rotenberg Section 2.1 Trace-Level Sequencing "...The trace predictor and trace cache together provide fast trace-level sequencing...Instruction-level sequencing, discussed in the next section, is required to construct nonexistent traces or repair trace mispredictions."). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the trace caches, buffers, and selector of Rotenberg in the device of Kyker to increase the speed of trace-level sequencing while using instruction-level sequencing to create new or correct mispredicted traces.

29. Referring to claim 16, Kyker in view of Rotenberg has taught the processing engine of claim 15, wherein the instruction segments are extended blocks (Kyker column 1, lines 30-45 "...the target address, the backward taken branch, and any micro-ops between the two form a loop..."; column 2, line 59 to column 3, line 33 "...Exemplary trace T1 includes a total of four micro-ops...The second exemplary trace T2 includes the same loop, L_H, L_I, L_2, but does not include any micro-op preceding the loop itself..."; Figure 1; and Figure 2 – In regards to Kyker, the loop contains a plurality of instructions, i.e. block of instructions, with four instructions and includes the extra micro-op preceding the loop itself.).

30. Referring to claim 17, Kyker in view of Rotenberg the processing engine of claim of 15, wherein the instruction segments are traces (Kyker Abstract "...a cache unit, which includes a data array that stores traces...In one exemplary method of unrolling loops, the processor or trace

cache unrolls loops…"; column 2, line 59 to column 3, line 33 "…when the trace cache determines that a loop is present, the trace cache continues to build the trace by building additional iterations of the loop until the trace is a minimal length…In other words, the trace cache builds the loop repeatedly until the trace is, for example, over two trace lines long…"; Figure 1; and Figure 2).

31. Referring to claim 18, Kyker in view of Rotenberg has taught the processing engine of claim 15, wherein the instruction segments are basic blocks (Kyker column 1, lines 30-45 "…the target address, the backward taken branch, and any micro-ops between the two form a loop…"; column 2, line 59 to column 3, line 33 "…Exemplary trace T1 includes a total of four micro-ops…The second exemplary trace T2 includes the same loop, L_H, L_I, L_2, but does not include any micro-op preceding the loop itself…"; Figure 1; and Figure 2 – In regards to Kyker, the loop contains a plurality of instructions, i.e. block of instructions, with three instructions and does not include the extra micro-op preceding the loop itself.).

32. Referring to claim 19, Kyker has not taught the processing engine of claim 15, wherein the instruction segment cache system further comprises a segment predictor provided in communication with the segment cache. Rotenberg has taught the processing engine of claim 15, wherein the instruction segment cache system further comprises a segment predictor provided in communication with the segment cache (Rotenberg Section 2.1 Trace-Level Sequencing "…A *next trace predictor*[14] treats traces as basic units and explicitly predicts sequences of traces…" and Figures 2). A person of ordinary skill in the art at the time the invention was made, and as taught by Rotenberg, would have recognized that the trace predictor achieves high branch prediction throughput with a single prediction per cycle (Rotenberg Section

2.1 Trace-Level Sequencing "...high branch prediction throughput is implicitly achieved with only a single trace prediction per cycle..."). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the trace predictor of Rotenberg in the device of Kyker to achieve high branch prediction throughput in a single prediction cycle.

Response to Arguments

33. Applicant's arguments, see Appeal Brief, filed 18 December 2006, with respect to the rejection(s) of claim(s) 1-19 have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of the above rejections.

Conclusion

34. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

a. BALA et al., U.S. Patent Application Publication 2002/0104075, has taught building a trace for blocks of "hot" instructions, including those with backward branches.

b. Peled et al., U.S. Patent Number 6,073,216 and 6,076,144, have taught building traces for multiple path instructions.

c. Hsu et al., U.S. Patent Number 6,418,530, has taught tracing and selecting bundles of instructions.

d. Kyker et al., U.S. Patent Number 6,594,734, has taught using a trace cache.

e. Patel et al. "Critical Issues Regarding the Trace Cache Fetch Mechanism" ©1997 has taught trace caches and their functions.

35. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J. Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:00am-4:30pm.

36. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

37. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Aimee J Li
Examiner
Art Unit 2183

15 April 2007

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100